1. (1) False.
   (2) False.
   (3) True.
   (4) True.
   (5) False.
   (6) True.
   (7) False.
   (8) False.
   (9) True.
   (10) True.

2. (a) 
   ```
   interface Person {
       attribute string name;
       attribute string birth\_date;
       relationship Person motherOf
           inverse Person::childrenOfFemale
       relationship Person fatherOf
           inverse Person::childrenOfMale
       relationship Set<Person> children
           inverse Person::parentsOf
       relationship Set<Person> childrenOfFemale
           inverse Person::motherOf
       relationship Set<Person> childrenOfMale
           inverse Person::fatherOf
       relationship Set<Person> parentsOf
           inverse Person::children
   }
   ```

   Notice that there are six different relationships here. For example, the inverse of the relationship that connects a person to their (unique) mother is a relationship that
connects a mother (i.e., a female person) to the set of her children. That relationship, which we call childrenOfFemale, is different from the children relationship, which connects anyone – male or female – to their children.

(b.i) See Figure 1.
(b.ii) See Figure 2.

3. (a.i) Check all three constraints.
(a.ii) Referential constraints.
(a.iii) None.
(b) No. Because \((AC)^+ = \{A, B, C, D, H\}\) and \(G \notin (AC)^+\).
(c) 1st NF. Because \(AB\) is the only key and \(C\) is partially dependent on \(A\).
(d) \(\{Y \rightarrow X, Z \rightarrow Y, Z \rightarrow W\}\).

4. (a.i) A primary index is an index on a set of fields that includes the primary key. It is guaranteed not to contain duplicates. A secondary index is an index that is not a primary index. It can have duplicates.
(a.ii) A dense index has at least one data entry search key value that appears in a record in the indexed file. A sparse index contains an entry for each page of records in a
data file. It must be clustered. Thus, we can have at most one sparse index on a data file. A sparse index is typically much smaller than a dense index.

(a.iii) A clustered index is one in which the ordering of data entries is the same as the ordering of data records. We can have at most one clustered index on a data file. An unclustered index is an index that is not clustered. We can have several unclustered indexes on a data file.

(b) No, it is not a good idea, because it does not provide useful indexing information for retrieval and is useless.

(c) No. In this case, the index is unclustered, each qualifying data entries could contain an aid that points to a distinct data page, leading to as many data page I/O as the number of data entries that match the range query. In this case, if the number of data entries is larger than the number of data blocks then using index is worse than file scan.

5. (a) See my lecture notes for such an example.

(b.i) A naive way to do this is to construct a serial schedule using the two-phase locking protocol.

(b.ii) No, it is impossible to have a non-serial but serializable schedule. Consider that the last operation in $T_1$ is WRITE (B), and the last operation in $T_2$ is WRITE (A). Meanwhile, $T_1$ starts with READ (A), and $T_2$ starts with READ (B). Therefore, in any serializable schedule of $T_1$ and $T_2$, either READ (A) in $T_1$ should be after WRITE (B) in $T_2$, or READ (B) in $T_2$ should be after WRITE (B) in $T_1$.

(b.iii) Yes. (Students must construct a schedule with deadlock. It is very straightforward to construct such a schedule, so I omit it here.)

(c) The schedule is not serializable, because there is a cycle in its schedule graph. See Figure 3.